Rapport du Mini-Projet AAI

Segmentation du Mouvement dans des Vidéos HAR

ConvLSTM Autoencoder & 3D-CNN

Étudiants :
Guettiche Zakaria
Djerafi Taki Eddine
Damous Anis

Encadrant : Dr. Baha Fergani

Année Universitaire : 2025–2026

Contents

1. Présentation du Domaine

1.1. Vision par Ordinateur

La vision par ordinateur est un domaine de l'intelligence artificielle qui vise à permettre aux machines de comprendre, analyser et interpréter des images ou des vidéos. Elle englobe plusieurs tâches telles que la classification, la détection d'objets, la segmentation, la reconnaissance d'activités et l'analyse du mouvement.

1.2. Segmentation d'Images et de Vidéos

La segmentation est le processus consistant à diviser une image ou une vidéo en régions homogènes ou cohérentes afin de simplifier ou transformer la représentation visuelle pour l'analyse. L'objectif est d'isoler des objets, des personnes, ou des zones d'intérêt.

1.3. Types de segmentation

1.3.1. Segmentation Sémantique

Assigne à chaque pixel une classe parmi un ensemble prédéfini (ex : personne, voiture, route).

Caractéristiques :

- Chaque pixel reçoit un label.
- Les objets d'une même classe ne sont pas distingués entre eux.

1.3.2. Segmentation d'Instances

Identifie non seulement la classe mais aussi l'identifiant unique de chaque occurrence d'objet.

Exemple : distinguer deux personnes distinctes dans une même image.

1.3.3. Segmentation Panoptique

Fusion de la segmentation sémantique et des instances.

Caractéristiques :

- Les surfaces amorphes (route, ciel) $\rightarrow$ segmentation sémantique.
- Les objets individuels (personnes, voitures) $\rightarrow$ instances séparées.

1.3.4. Segmentation du Mouvement

Permet d'isoler les régions de la vidéo qui changent d'un instant à l'autre.

Applications :

- Surveillance vidéo
- Analyse d'activité humaine

- Suivi d'objets
- Détection d'anomalies

1.4. Motivation du choix

Nous avons choisi la segmentation du mouvement car elle est adaptée à la détection d'activité humaine dans des vidéos, particulièrement lorsqu'on utilise un autoencodeur dont l'erreur de reconstruction met naturellement en évidence les zones en mouvement.

2. Problématique

La reconnaissance d'activité humaine repose sur la capacité à détecter les zones en mouvement dans une scène vidéo. Toutefois, plusieurs difficultés apparaissent :

- Variations rapides du mouvement humain.
- Présence d'objets statiques dans le fond.
- Qualité variable des vidéos du dataset.
- Ambiguïté entre certaines actions proches (ex : walking vs walking while reading).
- Besoin d'une méthode robuste sans annotation manuelle.

Notre objectif est de développer un modèle capable d'effectuer une **segmentation du mouvement** en exploitant un autoencodeur spatio-temporel (ConvLSTM), où :

- L'autoencodeur apprend à reconstruire les vidéos "normales".
- L'erreur de reconstruction met en évidence les zones qui changent.
- Un seuil transforme cette erreur en masque binaire segmenté.

3. Dataset

3.1. Dataset utilisé pour l'entraînement

3.1.1. Présentation du Dataset

- **Nom :** Human Activity Recognition (HAR - Video Dataset)
- **Source :** Kaggle
- **URL :** Dataset Kaggle – Human Activity Recognition
- **Type :** Vidéos RGB
- **Résolution d'origine :** variable
- **Nombre de classes :** 7

3.1.2. Répartition des classes

- Clapping : 146 vidéos
- Meet and Split : 147 vidéos
- Sitting : 156 vidéos
- Standing Still : 174 vidéos
- Walking : 171 vidéos
- Walking While Reading Book : 176 vidéos
- Walking While Using Phone : 143 vidéos

3.1.3. Motivation du choix

Dataset très populaire pour l'analyse d'activité humaine. Divers mouvement → idéal pour la segmentation du mouvement.

3.1.4. Défis

- Arrière-plan variable
- Mouvements rapides
- Qualité vidéo variable
- Actions similaires (walking / walking reading book)
- Jeux de données volumineux nécessitant des ressources importantes

3.2. Deuxième Dataset : Driving Video With Object Tracking

3.2.1. Présentation du Dataset

- **Nom :** Driving Video With Object Tracking
- **Source :** Kaggle
- **URL :** Driving Video With Object Tracking – Kaggle
- **Type :** Vidéos RGB (vidéos de conduite)
- **Résolution d'origine :** variable
- **Données fournies :** séquences vidéo + annotations de trajectoires/objets (tracking)
- **Tâches possibles :** détection/ suivi d'objets, segmentation d'objets en mouvement, analyse de scènes routières

3.2.2. Motivation du choix

Ce dataset est pertinent car il contient des vidéos réelles de conduite avec des objets variés et des annotations de suivi, offrant ainsi un ground truth pour la segmentation et la détection de mouvement, tout en permettant d'évaluer la généralisation du modèle dans un contexte différent.

3.2.3. Défis

- Arrière-plan variable
- Mouvements rapides
- beaucoup d'objets variés
- Jeux de données volumineux nécessitant des ressources importantes

4. Modèles Utilisés

4.1. ConvLSTM Autoencoder

4.1.1. Définition

Un ConvLSTM est une extension du LSTM où :

- Les opérations internes sont des convolutions 2D.
- Les états cachés conservent une structure spatiale.

4.1.2. Architecture détaillée

- Encoder : CNN + MaxPool
- Bottleneck : ConvLSTM
- Decoder : Convolution Transpose

4.1.3. Points forts

- Capture des dépendances spatio-temporelles
- Reconstruction fine des frames

4.1.4. Limites

- **Sensibilité au seuil de segmentation**
  Le paramètre *threshold* influence fortement le masque de segmentation final. Un choix inadéquat du seuil peut entraîner :
  - des faux positifs dus au bruit,
  - des faux négatifs, notamment pour les mouvements de faible amplitude.
- **Difficulté avec les mouvements lents**
  Les mouvements progressifs sont souvent bien reconstruits par l'autoencodeur, ce qui conduit à une erreur de reconstruction faible. Par conséquent, ces mouvements peuvent ne pas être détectés lors de la segmentation.
- **Absence de compréhension sémantique**
  Le modèle est basé uniquement sur l'erreur de reconstruction et détecte le mouvement sans comprendre la nature des objets. Il est ainsi incapable de distinguer entre
  - un humain
  - une ombre
  - ou un objet mobile.
- **Généralisation limitée**
  Les performances du modèle dépendent fortement des scènes utilisées lors de l'entraînement.

Un changement de caméra, d'angle de vue ou d'arrière-plan peut entraîner une dégradation notable des résultats.

- **Coût temporel**
  Le traitement séquentiel des frames par le ConvLSTM induit un temps de calcul relativement élevé. Cette approche est généralement plus lente que les modèles basés sur des architectures *3D-CNN* pures.

4.1.5. Hyperparamètres

- Learning rate : 10^{-4}
- Frames : 10
- Taille : 128×128
- Optimizer : Adam
- Loss : MSELoss

4.1.6. Organigramme de l'algorithme

L'algorithme de reconstruction et de segmentation des vidéos peut se résumer par les étapes suivantes :

1. **Chargement des vidéos :** récupération et conversion en tenseurs PyTorch, sélection de $T = 10$ frames par vidéo et redimensionnement à 128×128.
2. **Prétraitement :** normalisation des pixels entre 0 et 1 et conversion des frames au format [C,H,W].
3. **Encodage :** chaque frame est passée dans le CNN encoder pour extraire les features spatiales.
4. **Propagation temporelle :** les features extraites sont traitées par le ConvLSTM pour capturer les dépendances spatio-temporelles. Les états cachés h et c sont mis à jour à chaque frame.
5. **Décodage :** chaque feature temporelle est décodée par le CNN decoder pour reconstruire les frames.
6. **Calcul de la perte et optimisation :** comparaison des frames reconstruites avec les frames originales via la MSELoss, et mise à jour des poids avec Adam optimizer.
7. **Segmentation du mouvement (Post-traitement) :** calcul de l'erreur absolue pixel par pixel, moyenne sur les canaux RGB pour obtenir un masque grayscale, et application d'un seuil pour obtenir le masque binaire des zones en mouvement.

4.1.7. Motivation du choix de l'algorithme

Le choix d'un ConvLSTM autoencodeur pour ce projet est motivé par plusieurs raisons :

- **Capturer les dépendances spatio-temporelles :** contrairement à un autoencodeur classique qui traite chaque image indépendamment, le ConvLSTM prend en compte la dynamique temporelle des vidéos grâce à ses états cachés récurrents.
- **Segmentation non-supervisée :** le modèle apprend à reconstruire les vidéos, ce qui permet d'extraire les zones en mouvement à partir de l'erreur de reconstruction,

sans avoir besoin d'annotations manuelles.

- **Simplicité et efficacité :** l'architecture combine un encoder CNN pour l'information spatiale, un ConvLSTM pour la temporalité, et un decoder CNN pour la reconstruction, ce qui offre un compromis performant entre complexité et capacité de modélisation.
- **Adaptabilité :** ce modèle peut être utilisé pour différentes tâches vidéo (détection de mouvement, segmentation d'objets en mouvement, analyse d'activités humaines) sans modification majeure.

4.2. 3D-CNN Autoencoder

4.2.1. Définition

Un 3D-CNN Autoencoder applique des convolutions sur les dimensions spatiales et temporelles simultanément (height, width, time).

4.2.2. Avantages

- Capture directe des patterns spatio-temporels
- Simple à entraîner
- Architecture simple et stable

4.2.3. Limites

- Pas de mémoire temporelle longue
- Très sensible à la taille de la séquence
- Coût mémoire élevé si T ou H×W augmente
- Moins efficace pour actions complexes ou longues

4.3. Modèles compatibles

U-Net 3D, I3D, SlowFast

4.4. Transfert Learning / Fine Tuning

Dans ce travail, ni le transfert learning ni le fine tuning n'ont été utilisés. En effet, le modèle repose sur une approche auto-supervisée basée sur la reconstruction, où l'objectif est d'apprendre à partir des données elles-mêmes sans nécessiter de labels. Dans ce contexte, l'utilisation d'un modèle préentraîné n'est pas pertinente, car la tâche (prédire et reconstruire des images) diffère des tâches classiques pour lesquelles les modèles préentraînés sont conçus.

Différence entre les deux techniques :

- **Transfert Learning :** utilisation d'un modèle préentraîné sur une grande base de données pour extraire des caractéristiques générales, sans modifier ses poids.
- **Fine Tuning :** ajustement (partiel ou complet) des poids d'un modèle préentraîné afin de l'adapter précisément à une nouvelle tâche.

5. Prétraitement et Post-traitement

Prétraitemen

Un prétraitement a été appliqué sur les vidéos avant l'entraînement du modèle. La procédure comprend les étapes suivantes :

- **Extraction d'un nombre fixe de frames** à partir de chaque vidéo (10), en les répartissant uniformément sur toute la durée.
- **Conversion des frames en format RGB** afin d'unifier l'espace colorimétrique.
- **Redimensionnement des images** à une taille fixe (128×128) pour garantir une entrée cohérente au modèle.
- **Normalisation des pixels** en divisant par 255 afin de stabiliser l'apprentissage et améliorer la convergence.
- **Conversion en tenseurs PyTorch** avec permutation des dimensions.

Ce que nous avons remarqué : Le prétraitement a permis d'obtenir des données homogènes, facilitant l'entraînement du modèle. La normalisation a également contribué à une meilleure stabilité du loss et à une convergence plus rapide.

Post-traitemen

Un post-traitement simple a été appliqué après la prédiction :

- **Calcul de l'erreur de reconstruction** entre les frames originales et les frames reconstruites (MSE).
- **Visualisation de la carte d'erreur** permettant d'identifier clairement les zones en mouvement.

Ce que nous avons remarqué : Les zones présentant une erreur élevée correspondent effectivement aux parties en mouvement dans la vidéo. Cela confirme que l'autoencodeur apprend principalement la structure statique de la scène et met en évidence les variations temporelles, ce qui est cohérent avec l'objectif de segmentation du mouvement.

6. Résultats et Discussion

6.1. Métriques utilisées

Pour évaluer la performance du modèle auto-supervisé de segmentation de mouvement, nous avons utilisé des métriques basées sur la reconstruction vidéo, adaptées à un contexte sans ground-truth.

- **MSE (Mean Squared Error) :** mesure l'erreur moyenne au carré entre les frames originales et les frames reconstruites par le modèle. Elle permet d'évaluer la qualité globale de la reconstruction pixel par pixel.
- **PSNR (Peak Signal-to-Noise Ratio) :** métrique exprimée en décibels qui quantifie le rapport entre la puissance du signal original et le bruit de reconstruction. Plus le PSNR est élevé, meilleure est la qualité de reconstruction.
- **SSIM (Structural Similarity Index) :** mesure la similarité perceptuelle entre les frames originales et reconstruites, en prenant en compte la luminance, le contraste et la structure. Elle reflète mieux la qualité visuelle perçue.

Ces métriques offrent une évaluation complète du modèle : MSE pour la précision locale pixel-wise, PSNR pour la qualité globale, et SSIM pour la préservation des structures visuelles dans la vidéo.

6.2. Comparaison avec l'état de l'art

Méthode	**MSE**	**PSNR**	**SSIM**
PredRNN++	46	Non reporté	0.88
FPNet-OF	Non reporté	30.8 dB	0.95
PREDRNN	56.8	Non reporté	0.781
ConvLSTM Autoencoder	81.65	24 dB	0.712

6.3. Discussion

Résultats inférieurs à l'état de l'art à cause :

- Réduction de la résolution à 128×128
- Limitation de GPU Kaggle
- Autoencodeur non spécialisé segmentation

7. Ressources Matérielles

7.1. Configuration matérielle

Pour l'entraînement et l'expérimentation du modèle, les ressources suivantes ont été utilisées :

- **GPU :** NVIDIA Tesla P100
- **CPU :** Kaggle (vCPU fourni par la plateforme)
- **RAM :** Kaggle (environ 16 Go disponibles)

7.2. Contraintes et ajustements

Lors des tests initiaux, nous avons rencontré un problème de mémoire (Out Of Memory) lorsque :

- la taille des frames était de 256×256
- le nombre de frames par vidéo était $T = 60$

Pour résoudre ce problème et assurer un entraînement stable, les paramètres ont été ajustés comme suit :

- taille des frames : 128×128
- nombre de frames par séquence : $T = 10$

7.3. Apport du mini-projet

Ce mini-projet a permis de mettre en pratique et d'explorer les concepts suivants :

- **Segmentation non-supervisée** : détection des zones de mouvement à partir de l'erreur de reconstruction.
- **Reconstruction temporelle profonde** : apprentissage de représentations vidéo à l'aide d'un autoencodeur ConvLSTM pour modéliser la dynamique temporelle.

7.4. Algorithmes alternatifs pour pallier les limites du ConvLSTM Autoencoder

Pour résoudre certaines limitations du ConvLSTM Autoencoder, notamment la forte consommation mémoire et la sensibilité à la longueur des séquences, plusieurs approches alternatives ont été proposées dans la littérature :

- **PredRNN / PredRNN++ :** Ces modèles améliorent la prédiction vidéo en utilisant des architectures récurrentes spatio-temporelles plus efficaces et stables, réduisant les problèmes liés à la longueur des séquences et capturant mieux les dépendances temporelles longues.

- **ST-GCN (Spatial-Temporal Graph Convolutional Network) :** Utilisé pour la modélisation des dépendances spatio-temporelles dans les séquences vidéo, il permet de représenter les interactions entre objets et de diminuer la complexité de traitement par rapport aux ConvLSTM classiques.

Ces algorithmes permettent ainsi de pallier les limitations du ConvLSTM Autoencoder en offrant une meilleure efficacité mémoire et une meilleure capture des relations spatio-temporelles sur de longues séquences.

8. Conclusion

Le modèle ConvLSTM Autoencoder développé pour la segmentation du mouvement dans les vidéos donne de meilleurs résultats sur des séquences où le fond est relativement stable et uniforme, les objets en mouvement sont clairement visibles et de taille suffisante, et les mouvements de caméra ne sont pas trop brusques, ce qui limite les artefacts de reconstruction. Ce modèle peut être appliqué dans plusieurs domaines, tels que la surveillance vidéo et la détection d'anomalies, l'analyse d'activité humaine et la reconnaissance de gestes, le suivi d'objets pour les véhicules autonomes, ainsi que des applications multimodales combinant vision et analyse temporelle pour la robotique et la domotique.